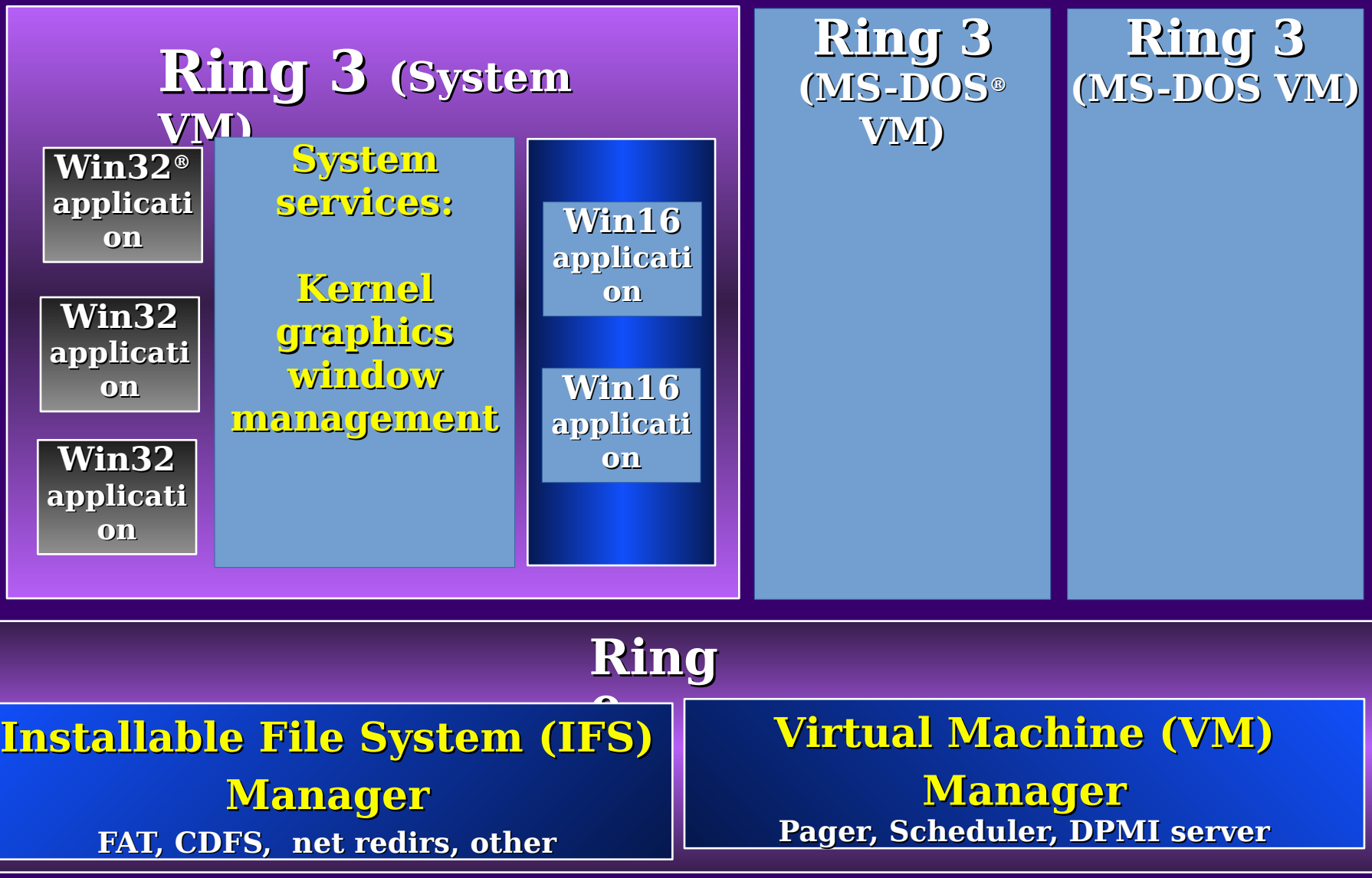
VxD Architecture For Windows™ *"Chicago"*

Ralph Lipe
Architect For *"Chicago"*
Microsoft Corporation

Agenda

- OS driver introduction
- Driver layout
- OS services
- Application/driver interactions
- Debugging
- Call to action
- More information

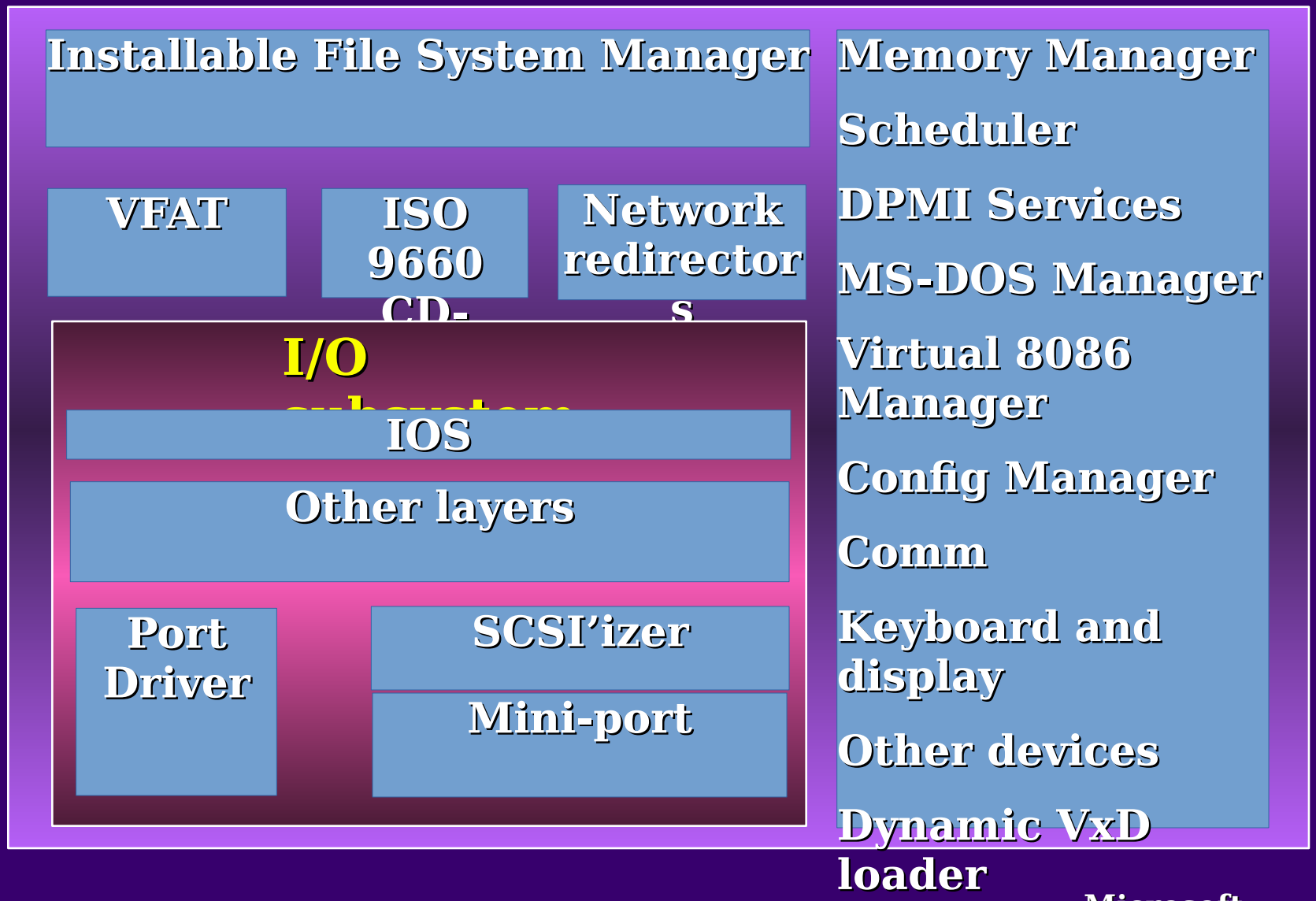
Core Architecture



Layered Drivers

- Most drivers have a layered driver model
  - Storage
  - Display
  - Printer
  - Network
  - Multimedia
  - Etc.
- There are PDC presentations for these. This presentation is not for a particular driver model

Core Components



System Memory Map

OS Core Components

- Runs in 32-bit, flat, i386 ring 0 protect mode
- MS-DOS-based applications run in virtual machines (VMs)
- All Windows-based applications run in same VM
- Virtual Machine Manager (VMM)
  - Controls VMs/threads
  - Scheduler
  - Memory management

OS Core Components

- I/O Supervisor (IOS)
  - Controls disk I/O layers
- Installable File System Manager (IFSMgr)
  - Controls file system drivers (FSDs)
- Configuration Manager
  - Controls Plug and Play events

VMs, Threads, And Applications

- System VM
  - Each Win32 process resides here
    - Each has a separate address space
    - Each can have multiple threads
  - All Win16 processes reside here
    - All share same address space
    - Each one is a single synchronized thread
- MS-DOS VMs
  - One for each MS-DOS-based application running
    - Each one is a single thread

VxDs

- VxD is an acronym for virtual (x) driver
- Originally for virtualizing hardware for VMs
- Interfaces provided for C or MASM
- Runs in 32-bit, flat, ring 0 protect mode
- VxDs can provide services for other drivers

Driver Code/Data Layout

- Fully pageable (code and data)
  - Minimize locked code to save system memory
  - Windows 3.x VxDs could only page allocated memory
- Message macros
  - Easier to localize data
- Code/data segment macros...

Driver Segment Layout

- VxD_<x>_SEG, VxD_<x>_ENDS macros
  - ICODE & IDATA: initialization
  - CODE & DATA: resident
  - LOCKED_CODE & _DATA: resident locked
  - PAGEABLE_CODE & _DATA: swappable
  - STATIC_CODE & _DATA: DL-VxD static
  - DEBUG_ONLY_CODE & _DATA: debug-only

Device Declaration

```
Declare_Virtual_Device
    VDD,
    2, 0,
    VDD_Control,
    VDD_Device_ID,
    VDD_Init_Order,
    VDD_PM_API
```

Device IDs

- If you export services you need an OEM ID
- With an OEM ID you can create up to 16 different VxDs
- 0-1FFFh reserved for standard devices
- If you do not have one, contact us!
  - Electronic mail: vxdid@microsoft.com

Real-Mode Initialization

- Static VxDs can run part of their init code in real mode, prior to entering protected mode
- New services available for read-only registry access

System Control Messages

- System initialization/termination
- VM manipulation
- Thread manipulation
- Advanced power management (APM)
- Plug and Play
- Win32 DeviceIoControl entry

System Control Messages

- Static VxDs only need to receive:
  - SYS_CRITICAL_INIT
  - DEVICE_INIT
  - INIT_COMPLETE
- Dynamic VxDs only need to receive:
  - SYS_DYNAMIC_DEVICE_INIT
  - SYS_DYNAMIC_DEVICE_EXIT
- VxDs that receive superset can be loaded as either static or dynamic

Device Control Procedure

```
Begin_Control_Dispatch VDD
        Control_Dispatch Set_Device_Focus,  VDD_Set_Dev_Focus
        Control_Dispatch Begin_Message_Mode,VDD_Begin_Msg_Mode
        Control_Dispatch End_Message_Mode,  VDD_End_Msg_Mode
        Control_Dispatch VM_Suspend,        VDD_VM_Suspend
        Control_Dispatch VM_Resume,         VDD_VM_Resume
        Control_Dispatch Create_VM,         VDD_Create_VM
        Control_Dispatch Destroy_VM,        VDD_Destroy_VM
        Control_Dispatch VM_Critical_Init,  VDD_VM_Crit_Init
        Control_Dispatch VM_Init,           VDD_VM_Init
        Control_Dispatch Sys_VM_Init,       VDD_Sys_VM_Init
        Control_Dispatch System_Exit,       VDD_System_Exit
        Control_Dispatch Device_Init,       VDD_Device_Init
        Control_Dispatch Sys_Critical_Init, VDD_Sys_Crit_Init
        Control_Dispatch Init_Complete,     VDD_Init_Complete
        Control_Dispatch Sys_Critical_Exit, VDD_Sys_Crit_Exit
IFDEF   DEBUG
        Control_Dispatch Debug_Query,       VDD_DebugQuery
ENDIF
End_Control_Dispatch VDD
```

Loading VxDs

- Statically loadable VxDs
  - Both real-mode and protect-mode init
  - Driver stays loaded, cannot unload
- Dynamically loadable VxDs (DLVxDs)
  - No real-mode init, all protect-mode
  - Static data available to survive unload/reload
  - New VMM services to undo things
  - Many VMM services are no longer init-time-only, to allow for

How Drivers Get Called

- Services you export
  - Calls to your exported VxD services
    - Other VxDs can do this
- System control messages
  - Other VxDs can do this
  - Win32 DeviceIoControl does this
- Entry point
  - MS-DOS-based and Win16 applications can do this
  - Win32 applications do a device IOCTL

Exporting/Importing Services

```
Begin_Service_Table VDD
VDD_Service VDD_Get_Version
VDD_Service VDD_PIF_State
VDD_Service VDD_Get_GrabRtn
VDD_Service VDD_Hide_Cursor
VDD_Service VDD_Set_VMType
VDD_Service VDD_Get_ModTime
VDD_Service VDD_Set_HCurTrk
VDD_Service VDD_Msg_ClrScrn
VDD_Service VDD_Msg_ForColor
VDD_Service VDD_Msg_BakColor
VDD_Service VDD_Msg_TextOut
VDD_Service VDD_Msg_SetCursPos
VDD_Service VDD_Query_Access
; New svcs for 3.1.  Not supported by 3.0 VDD's
VDD_Service VDD_Check_Update_Soon,,VGA31
; New services for DOS386.
VDD_Service VDD_Fullscreen_Grab,,DOS386
End_Service_Table VDD
```

VMM Services

- Registry
- Scheduling
  - Primary scheduler
  - Time slice scheduler
  - Mutexes
  - Semaphores
- Events
- Timer
- I/O trapping
- Nested execution

VMM Services

- Memory management
  - Heap allocation
  - Page allocation and management
  - Device V86 page management
  - Free physical memory management
  - GDT/LDT memory management
  - Protect-mode memory management
  - V86 address space mapping and allocation

VMM Services

- Processor fault/interrupt
- Protected-mode execution
- Breakpoint/callback
- VM interrupt/callback
- Information
- Initialization information
- Linked list
- Error condition

VMM Scheduler Services

- Thread-based preemptive scheduler
- New synchronization objects
  - Semaphores
  - Mutexes
- Compatible with Windows NT™ model: 32 priority levels
- Also compatible with existing Win16 applications, VDDs, and VxDs

VMM Scheduler Services

- Dynamic priority boosting with timed decay
- Priority inversion boosting
- VM priority boost
  - Boosts all system VM threads
- VxDs can obtain per-thread data

VMM Event Services

- Events with restrictions
  - Triggered when restrictions met
  - Example: event when no VPICD hardware interrupt simulation
- Events when Global condition met
  - Example: event when interrupts enabled
  - Can be in the context of any thread

VMM Event Services

- "Appy time" event
  - Guarantees application context
  - Allows Ring 3 dispatch to specific DLL/application
    - Shell services provide Win32 API access
  - Broadcast: reaches all applications, drivers, VxDs
    - Provides 2-way communication
    - Example: used for dock/undock

VMM Registry Services

- Hierarchical database
- Parallels Ring 3 Win32 API
- LDRSRV_Reg* services for static VxDs
  - For use during real-mode initialization
  - Read-only access to HKEY_LOCAL_MACHINE

VMM Registry Services

```
cbMyKey = sizeof(szMyKey);
if (GetRegistryPath(&DDB, szMyKey, cbMyKey)) {
   if (RegOpenKey(HKEY_LOCAL_MACHINE, szMyKey, &hk)
                  == ERROR_SUCCESS) {
      cbValLen = sizeof(szValue);
      rr = RegQueryValueEx(hk, szValueName, NULL,
                           NULL, szValue, cbValLen);
      RegCloseKey(hk);
   }
}
```

Configuration Manager Services

- Device insertion and removal notifications
- APIs to get I/O, memory, IRQ, and DMA allocation
- Exact interfaces are class-specific
- See breakout session for the type of driver you are interested in for more details on Plug and Play

VDMAD DMA Services

- New VDS service: DMA buffer allocation
  - Allows DMA buffer allocation at any time, no longer only at system initialization
  - Necessary for dynamic VxDs
  - Avoid init time allocation

VDMAD DMA Services

- _PageAllocate(nPages,
  PG_SYS, 0, AlignMask, 0,
  1000h, &dwPhysAddr,
  PAGEUSEALIGN |
  PAGECONTIG |
  PAGEFIXED);

VTD Timer Services

- VTD (virtual timer device) now tracks time more accurately
- System now based off RTC, not line clock
  - System doesn't have to trap I/O ports to attempt to recover from applications and drivers who play with timer chip
- Async time-outs
  - Called at timer_int_time

Shell DeviceBroadcast Services

- Sample WM_DEVICEBROADCAST event codes:
  - DBT_DRIVEADD
    - The drive has been added
  - DBT_DRIVEREMOVED
    - The drive has been removed
  - DBT_DRIVEQUERYREMOVE
    - Testing removal of the drive
  - DBT_DRIVEQUERYREMOVEFAILED
    - Testing removal of the drive failed

Other Services From Core VxDs

- VPICD: PIC Services
- VxDLdr: VxD Loader Services
- Shell: Shell Services
- DOSMgr: MS-DOS Manager Services
- V86MMgr: V86 Memory Manager Services
- Enable: Accessibility and Disabilities Services

Debugging

- Debug kernel
  - Provided with DDK
  - Performs parameter validation
  - Always test with this during development, later use retail kernel
- Debuggers for VxDs
  - WDeb386.exe
    - Requires serial terminal
  - Nu-Mega's Soft-ICE/W

Calls To Action!

- Design Plug and Play hardware
  - ISA-PnP, PCMCIA, PCI
- Write Plug and Play drivers
  - Support DBT_* messages
- Write dynamically loadable VxDs
  - Only load when necessary

More Calls To Action!

- Use localization macros for messages
- Use MapPhysToLinear, don't assume mapping
- Interrupt latency < 200 microseconds
  - Avoid turning interrupts off for long
- Avoid allocating DMA buffers during init time
  - Only allocate buffers when

More Information

- Many other PDC presentations!
- “Chicago” Device Driver Kit
- CompuServe® support forums
  - WinDDK (“Chicago” DDK)
  - PlugPlay (Plug and Play)
- Microsoft Developer Network (MSDN)
  - Level 1: Development Library and Newsletter
  - Level 2: Development Platform